Idomaar: A Framework for Multi-dimensional Benchmarking of Recommender Algorithms

Mario Scriminaci[1], Andreas Lommatzsch[2], Benjamin Kille[2], Frank Hopfgartner[3], Martha Larson[4], Davide Malagoli[1], Andras Sereny[5], Till Plumbaum[2]
[1]ContentWise R&D – Moviri, Milan, Italy, {firstname.lastname}@moviri.com
[2]TU Berlin – DAI-Lab, Berlin, Germany, {firstname.lastname}@dai-labor.de
[3]University of Glasgow, Glasgow, UK, frank.hopfgartner@glasgow.ac.uk
[4]TU Delft, Delft, The Netherlands, m.a.larson@tudelft.nl
[5]Gravity R&D, Budapest, Hungary, sereny.andras@gravityrd.com

ABSTRACT

In real-world scenarios, recommenders face non-functional requirements of technical nature and must handle dynamic data in the form of sequential streams. Evaluation of recommender systems must take these issues into account in order to be maximally informative. In this paper, we present Idomaar—a framework that enables the efficient multi-dimensional benchmarking of recommender algorithms. Idomaar goes beyond current academic research practices by creating a realistic evaluation environment and computing both effectiveness and technical metrics for stream-based as well as set-based evaluation. A scenario focussing on "research to prototyping to productization" cycle at a company illustrates Idomaar's potential. We show that Idomaar simplifies testing with varying configurations and supports flexible integration of different data.

1. INTRODUCTION AND MOTIVATION

Increasingly, we witness a shift of recommender system research toward large-scale systems developed for industry settings. The trend was already well described in Amatriain's 2012 tutorial on building large-scale real-world recommender systems at ACM RecSys 2012 [1]. Given commercial systems' complexity and the demand for high performance, evaluation is subject to additional requirements: contribution of complementary information, reliablility on handling large-scale problems, and use of different methods and metrics. Evaluation must allow both offline parameter tuning as well as monitoring systems online.

Benchmarking the performance of recommender systems by these aspects is challenging. Mark Levy pointed this out during his keynote at the ACM RecSys 2013 workshop on Reproducibility and Replication in Recommender Systems [6]. Said and Bellogín [7] concur with his point as they analyze existing frameworks' abilities. Many commonly used software suites do not provide the required functionalities to benchmark different aspects, or they are too complex to set up.

We introduce *Idomaar* to address this challenge.[1] It enables researchers to evaluate different algorithm with respect to multiple criteria. The framework uses large-scale static data sets to simulate live data streams, bringing offline evaluation closer to online A/B testing. By comparing the performance of recommender algorithms operating in a live system (e.g., as studied in the living lab NewsREEL [2]) and these simulated data streams, the framework can be used to study the transferability of offline evaluation to an online setting. Finally, Idomaar enables *multi-dimensional* evaluation which simultaneously measures the performance of algorithms with respect to precision-related and technical aspects. These cover CTR and scalability-related measures (throughput and response time).

[1]Idomaar is available at `https://github.com/crowdrec/idomaar` see also `http://rf.crowdrec.eu`

RecSys 2016 Poster Proceedings, September 15-19, 2016, Boston, USA .

2. APPROACH AND FRAMEWORK

The reference framework *Idomaar* is a tool to evaluate recommendation services in real-world settings. As opposed to typical recommender system evaluation which assumes static information, real-world applications process data in form of a stream of information. In fact, users, items, and interaction amid both collections continue generating events fed to the recommender system. For instance, new users register or existing users cancel their subscription; new items emerge; users consume items. Such information must be ingested and processed as soon as possible (e.g., by updating the recommendation models) in order to be available. All these messages are asynchronously handled. However, the system also has to **synchronously** serve incoming recommendation requests within strict time constraints. Practically, the whole flow of incoming *messages* is managed by means of queues.

Idomaar mimics the work flow of such real-world scenario by using state-of-the-art technologies (e.g., Apache Flume and Apache Kafka) to manage data streaming. The architecture has been split into four main modules, as depicted in Fig. 1: the data container, the evaluator, the orchestrator, and the computing environment.

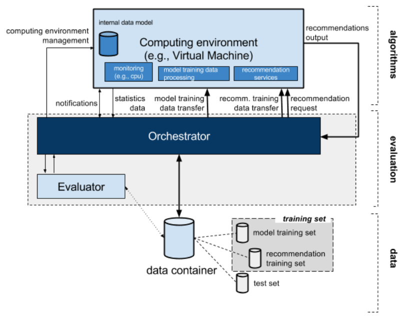

Figure 1: The Idomaar reference framework architecture.

The data container stores the data (entities and relations, in accordance to the format defined in [9]. Part of the data bootstraps the recommender system for training algorithms, while most of the remaining data feeds the recommender system at real-time while it has to serve incoming recommendation requests for test purposes.

Finally, the remaining subset of the data, the ground truth, is hidden from the recommender system and used to evaluate the quality of the service in terms of the user metrics. The data is read by a custom Apache Flume source and sent into an Apache Kafka queue.

The recommender system runs within a virtual machine, referred to as *computing environment*, whose environment is created with Vagrant (`https://www.vagrantup.com/`) and where all required libraries are automatically provisioned with Puppet (`https://puppetlabs.com/`). The recommender system subscribes to the required Apache Kafka channel and receives the asynchronous messages (i.e., users, items, interactions, etc.). Recommendation requests are synchronously sent via a HTTP interface (or, alternatively, a 0MQ interface).

The evaluator compares recommendations generated by the computing environment with the ground truth. In addition to standard user metrics (RMSE, recall, precision), Idomaar evaluates business metrics (e.g., scalability, response time, throughput), so to provide a 360-degree evaluation of the recommendation infrastructure.

Finally, the orchestrator coordinates all processes, including launching and provisioning the computing environment, instructing the evaluator to split data into training, test, and ground truth, feeding the recommender system with the incoming messages in accordance to their timestamp, collecting the generated recommendations, and computing the quality metrics.

Moving from an offline toward an online scenario (where data stream is not simulated from historical information, but the real flow of data) means either replacing the Apache Flume source with another one (e.g., that reads from log data) or ingesting the data directly into the Apache Kafka queue.

3. RELATED WORK

Various frameworks have been proposed to facilitate evaluating recommender systems. Ekstrand et al. [3] introduce `LensKit` to increase comparability of recommender system evaluation. `Mahout` is a scalable machine learning toolkit implemented in `Java`. Both frameworks ship with a selection of recommendation algorithms and some evaluators. Gantner et al. [4] created `MyMediaLite` as a lightweight recommender system framework. It comprises some recommendation algorithms along with predefined evaluation protocols. Said and Bellogín [8] proposed `RiVal` to facilitate comparing various recommendation algorithms. The framework's architecture supports cross-framework comparisons. The variety in frameworks emphasizes the demand for tools to evaluate recommender systems. Although the presented tools support evaluation, all presented frameworks measure quality only in terms of predictive performance. Operating recommender systems face additional challenges. For instance, they might be subject to response time restrictions or experience heavy load. Finally, running above mentioned frameworks on different hardware still yields inconsistent results. For these reasons, we propose `Idomaar` a language-agnostic framework with cloud-support and the ability to measure time and space complexity.

4. PROTOTYPE TO PRODUCTIVIZATION

Idomaar was used in the "research to prototyping to operating" cycle of a recommender system service provider to validate its usefulness. The focus of the validation was on multidimensional evaluation that simultaneously takes effectiveness and technical constraints into account. The only limitation identified was the monitoring of performance metrics like CPU and memory usage (cf. [5, 10]).

With respect to the cycle itself, we found that having a standard in terms of data formats and APIs increases the reusability of code in all phases and helps data scientists to produce code that can be transformed into effective prototypes. Our "research to prototyping to operating" cycle shows:

- Idomaar allowed easy testing using different datasets with different algorithms that share the same input types and subjects (e.g., implicit or explicit events, sessions or users).
- The Idomaar format is flexible enough to change subjects, events types, or to integrate contextual information, both on events and on recommendation requests.
- Idomaar can be considered as a suited tool for recommender system research: reuse of code speeds up prototyping and standardization of datasets helps merging different data sources.

In the future, Idomaar will go beyond classical recommender systems domains (e.g., movies or products) and consider additional types such as actions or navigation trees. Supporting generic objects and additional evaluation functions promise to establish Idomaar as standard research tool for recommender system. Such a standard could provide valuable support for the current trend of researchers participating in community-wide recommender system challenges. Idomaar has already been applied in such a challenge.

5. CONCLUSION

In this paper, we present the Idomaar framework, which enables the efficient, reproducible evaluation of recommender algorithms in real-world stream-based scenarios. Idomaar simplifies the multi-dimensional evaluation taking into account precision-related metrics as well as technical aspects.

Acknowledgment: The research leading to these results was performed in the CrowdRec project, which has received funding from the EU 7th Framework Programme FP7/2007-2013 under grant agreement No. 610594.

6. REFERENCES

[1] X. Amatriain. Building industrial-scale real-world recommender systems. In *RecSys '12*, pages 7–8, 2012.

[2] T. Brodt and F. Hopfgartner. Shedding light on a living lab: the CLEF NEWSREEL open recommendation platform. In *IIiX '14*, pages 223–226, 2014.

[3] M. D. Ekstrand, M. Ludwig, J. A. Konstan, and J. T. Riedl. Rethinking the recommender research ecosystem: Reproducibility, openness, and lenskit. In *RecSys'11*, 2011.

[4] Z. Gantner, S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Mymedialite: A free recommender system library. In *RecSys'11*, pages 305–308. ACM, 2011.

[5] Y. Koh, R. Knauerhase, P. Brett, M. Bowman, Z. Wen, and C. Pu. An analysis of performance interference effects in virtual environments. In *ISPASS'07*. IEEE, 2007.

[6] M. Levy. Offline evaluation of recommender systems: all pain and no gain? In *RecSys 2013*, page 1, 2013.

[7] A. Said and A. Bellogín. Comparative recommender system evaluation: Benchmarking recommendation frameworks. In *RecSys'14*, RecSys '14, pages 129–136. ACM, 2014.

[8] A. Said and A. Bellogín. Rival: A toolkit to foster reproducibility in recommender system evaluation. In *RecSys'14*, pages 371–372, 2014.

[9] A. Said, B. Loni, R. Turrin, and A. Lommatzsch. An extended data model format for composite recommendation. In *RecSys'14 (Posters)*, 2014.

[10] O. Tickoo, R. Iyer, R. Illikkal, and D. Newell. Modeling virtual machine performance: challenges and approaches. *SIGMETRICS Perf. Evaluation Review*, 37(3):55–60, 2010.